EXAMEN: **Scriptingtalen**

1e Bachelor Informatica
prof. dr. Peter Dawyndt
groep 1

woensdag 15-06-2017, 14:00
academiejaar 2016-2017
eerste zittijd

Opgave 1: Binnenstebuiten (Python) (10 pt)

In morsecode wordt elk karakter van een woord vertaald naar een unieke opeenvolging van punten (.) en streepjes (-), en worden de vertalingen van de individuele karakters telkens van elkaar gescheiden door één enkele spatie. Als we in deze voorstelling even alle spaties wegdenken, dan kunnen we enkele nieuwe definities bedenken.

We zeggen dat woorden **homoniemen** zijn, indien ze dezelfde voorstelling in morsecode hebben als we daarin de spaties wegdenken. De volgende drie woorden zijn bijvoorbeeld homoniemen:

WOORD	MORSECODE
ADMITTED	.- -.. -- .. - - . -..
EMIGATED	. -- .. --. .- - . -..
PANACE	.--. .- -. .- -.-. .

Het **inverse** van een woord bekomen we door in de voorstelling in morsecode alle punten en streepjes om te wisselen: punten worden streepjes en streepjes worden punten. Uiteraard heeft niet elk woord een bestaand inverse, terwijl andere woorden dan weer meerdere homoniemen als inverse kunnen hebben. Hier zijn alvast enkele voorbeelden:

WOORD	MORSECODE	INVERSE MORSECODE	INVERS WOORD
ADMITTED	.- -.. -- .. - - . -..	- .. --. --- -.--	TIZZY
DEMENTED	-.. . -- . -. - . -..	.--- ... .-. -.--	JURY
FINNIEST	..-. .. -. -. .. . ... -	- -.-- -.-. --- --- -.	TYCOON
NINETEEN	-. .. -. . - . . -.	.- - - . -- .--. -	ATTEMPT
NOSTALGIA	-. --- ... - .- .-.. --. .. .-	.--. .- -- . -. - .- - .. --- -.	LAMENTATION

We zeggen dat een woord **palinmorse** is, als de voorstelling in morsecode van het woord palindromisch is. Hierbij laten we de spaties in die voorstelling buiten beschouwing. Hier zijn alvast enkele voorbeelden van palinmorse woorden:

WOORD	MORSECODE
ANNEXING	.- -. -. . -..- .. -. --.
INTERNAL	.. -. - . .-. -. .- .-..
PROTECTORATE	.--. .-. --- - . -.-. - --- .-. .- - .
SOPRANOS	... --- .--. .-. .- -. --- ...
WINTERTIME	.-- .. -. - . .-. - .. -- .

Opgave

In deze opgave maken we gebruik van de tekstbestanden `morsecode.txt` en `woorden.txt` om gegeven woorden om te zetten naar hun voorstelling in morsecode, en om alle homoniemen op te zoeken die corresponderen met een gegeven voorstelling in morsecode. Hierbij mag je er altijd van uitgaan dat

deze bestanden zich in de huidige directory bevinden. Elke regel van het tekstbestand `morsecode.txt` bevat een symbool bestaande uit één enkel karakter, gevolgd door een tab en de voorstelling van het symbool in morsecode. Alle symbolen die voorkomen in het bestand zijn verschillend. Het tekstbestand `woorden.txt` bevat een lijst van woorden, die elk op een afzonderlijke regel staan.

Definieer een klasse `Morsecode` die kan gebruikt worden voor het omzetten van woorden naar morsecode en het opvragen van alle homoniemen die corresponderen met een gegeven voorstelling in morsecode. Bij het instantiëren van objecten van de klasse `Morsecode` moeten de locaties van twee tekstbestanden doorgegeven worden: *i*) een bestand dat opgemaakt is volgens het formaat van het bestand `morsecode.txt` en *ii*) een bestand dat opgemaakt is volgens het formaat van het bestand `woorden.txt`. Voorts moet de klasse `Morsecode` de volgende methoden ondersteunen:

- Een methode `morse` waaraan een string moet doorgegeven worden. De methode moet de vertaling van de gegeven string naar morsecode teruggeven. Hierbij moet elk karakter van de string omgezet worden naar de corresponderende voorstelling van het symbool in morsecode uit het bestand dat als eerste werd opgegeven bij het instantiëren van het object. Deze voorstellingen in morsecode van de individuele karakters moeten dan samengevoegd worden, telkens van elkaar gescheiden door één enkele spatie. Indien de gegeven string karakters bevat die niet als symbool voorkomen in het bestand dat als eerste werd opgegeven bij het instantiëren van het object, dan moet de functie een `AssertionError` opwerpen met de boodschap `ongeldig woord`.
- Een methode `homoniemen` waaraan een voorstelling in morsecode moet doorgeven worden. De methode moet een verzameling teruggeven met alle woorden uit het bestand dat als tweede werd opgegeven bij het instantiëren van het object, waarvan de voorstelling in morsecode overeenkomt met de gegeven voorstelling in morsecode. Hierbij mag geen rekening gehouden worden met eventuele spaties in deze voorstellingen in morsecode.

Definieer ook nog een klasse `Morsewoord` waarmee alle woorden met een geldige voorstelling in morsecode kunnen voorgesteld worden (de zogenaamde **morsewoorden**). Dit zijn de woorden die voorkomen in het bestand `woorden.txt` én waarvan alle karakters als symbool voorkomen in het bestand `morsecode.txt`. Bij het instantiëren van objecten van de klasse `Morsewoord` moet een woord met een geldige voorstelling in morsecode doorgegeven worden. Indien dit niet het geval is, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldig morsewoord`. Voorts moet de klasse `Morsewoord` het volgende gedrag ondersteunen:

- De ingebouwde functies `repr` en `str` moeten kunnen gebruikt worden om stringvoorstellingen van de objecten op te vragen. Bekijk onderstaande voorbeeldsessie om de opmaak van deze stringvoorstellingen te achterhalen.
- De operator `==` moet kunnen gebruikt worden om na te gaan of twee morsewoorden gelijk zijn. Twee morsewoorden zijn gelijk als en slechts als ze homoniemen zijn.
- De unaire operator `-` moet kunnen gebruikt worden om het inverse morsewoord (een object van de klasse `Morsewoord`) te bepalen, dat als volgt bekomen wordt:
  - bepaal de voorstelling in morsecode van het morsewoord
  - wissel in deze morsecode alle punten en streepjes om: punten worden streepjes en streepjes worden punten
  - bepaal alle homoniemen die overeenkomen met deze nieuwe morsecode
  - het inverse morsewoord is het lexicografisch eerste van deze homoniemen

  Indien dit niet resulteert in een woord met een geldige voorstelling in morsecode, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldige bewerking`.

- De binaire operator + moet kunnen gebruikt worden om twee morsewoorden bij elkaar op te tellen. Het resultaat is een nieuw morsewoord (een object van de klasse `Morsewoord`) dat als volgt bekomen wordt:
  - bepaal de voorstelling in morsecode van de twee morsewoorden die bij elkaar opgeteld worden
  - voeg deze twee voorstellingen samen tot een nieuwe voorstelling in morsecode
  - bepaal alle homoniemen die overeenkomen met deze nieuwe morsecode
  - het resultaat is het lexicografisch eerste van deze homoniemen

  Indien dit niet resulteert in een woord met een geldige voorstelling in morsecode, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldige bewerking`.
- De klasse ondersteunt een methode `isPalinmorse` waaraan geen argumenten moeten doorgegeven worden. Deze methode moet een Booleaanse waarde teruggeven, die aangeeft of het morsewoord al dan niet palinmorse is.

Voorbeeld

In deze opgave gaan we er altijd van uit dat de huidige directory de tekstbestanden `morsecode.txt` en `woorden.txt` bevat.

```
>>> code = Morsecode('morsecode.txt', 'woorden.txt')
>>> code.morse('ADMITTED')
'.- -.. -- .. - - . -..'
>>> code.morse('EMIGATED')
'. -- .. --. .- - . -..'
>>> code.morse('PANACE')
'.--. .- -. .- -.-. .'
>>> code.homoniemen('.- -.. -- .. - - . -..')
{'ADMITTED', 'EMIGATED', 'PANACE'}

>>> woord = Morsewoord('ADMITTED')
>>> woord
Morsewoord('ADMITTED')
>>> print(woord)
.- -.. -- .. - - . -..
>>> Morsewoord('ADMITTED') == Morsewoord('PANACE')
True
>>> -woord
Morsewoord('TIZZY')
>>> woord.isPalinmorse()
False
>>> Morsewoord('SOPRANOS').isPalinmorse()
True
>>> Morsewoord('ABIDE') + Morsewoord('AMASS')
Morsewoord('PESTHOLES')

>>> Morsewoord('PANIC!')          # uitroepteken heeft geen voorstelling in morse
Traceback (most recent call last):
AssertionError: ongeldig morsewoord

>>> Morsewoord('FACEBOOK')        # woord komt niet voor in woordenlijst
Traceback (most recent call last):
AssertionError: ongeldig morsewoord

>>> -Morsewoord('ANDROID')        # geen woord met inverse morsecode in woordenlijst
Traceback (most recent call last):
AssertionError: ongeldige bewerking
```

Opgave 2: Kwyjibo (Python) (10 pt)

In de episode *Bart the Genius* van de Simpsons is het gezin een spelletje Scrabble aan het spelen, waarbij Bart (de zoon) het zelfverzonnen woord `kwyjibo` op het spelbord legt. Wanneer Homer (de vader) hem er weinig subtiel op wijst dat dit woord niet bestaat, komt Bart met een definitie op de proppen (met een addendum van Marge, de moeder):

> *kwyjibo: a big, dumb, balding North American ape with no chin (and a short temper)*

Homer kan er helemaal niet mee lachen, meer nog omdat het een onverholen omschrijving is van zijn eigen persoonlijkheid. Wanneer hij met een banaan in de hand op zijn zoon afstormt

> *I'll show you a big, dumb, balding ape!*

schreeuwt Bart het uit

> *Uh oh. Kwyjibo on the loose!*

Opgave

Het bordspel Scrabble wordt gespeeld op een bord bestaande uit $n \times n$ vakjes die gerangschikt zijn in een vierkant rooster met n rijen en n kolommen. In elk vakje past een blokje waarop een letter of een blanco staat. De spelers trekken een vast aantal blokjes uit een stoffen zak, en moeten daar om de beurt woorden mee vormen die ze op het spelbord aanleggen. De woorden mogen zowel van links naar rechts als van onder naar boven gespeld worden.

De kolommen van het bord worden van links naar rechts gelabeld met de opeenvolgende hoofdletters van het alfabet, en de rijen worden van onder naar boven gelabeld met de natuurlijke getallen 1, 2, 3, ... We gaan er voor de eenvoud van uit dat $1 \leq n \leq 26$. De plaats waar een woord op het spelbord gelegd wordt, wordt aangeduid aan de hand van twee strings:

- een string xy geeft de startpositie en de richting op het bord aan, waarbij x het label van de rij of de kolom aanduidt waarop het woord gespeld wordt, en y de tweede coordinaat aanduidt van de eerste letter van het woord
- een string met de letters van het woord; alle letters worden voorgesteld door hoofdletters, behalve als er een blanco blokje gebruikt wordt voor een bepaalde letter; in dat laatste geval wordt de gekozen letter aangeduid door de kleine letter variant

Op die manier geeft `3C WEB` aan dat het woord `WEB` horizontaal wordt aangelegd op de derde rij, en dit vanaf de derde kolom. Analoog geeft `C2 AWnING` aan dat het woord `AWNING` verticaal wordt aangelegd

in de derde kolom, te beginnen vanaf de tweede rij. Bovendien werd voor de derde letter van het woord `AWNING` een blanco gebruikt, waarvoor de letter n werd vastgelegd.

Als een woord op het spelbord aangelegd wordt, moet rekening gehouden worden met de letters die reeds op het bord liggen (inclusief de letters die gekozen werden bij het aanleggen van blanco blokjes). Hiervoor wordt een notatie gebruikt waarbij de letters die reeds op het bord lagen tussen ronde haakjes gezet worden. In bovenstaand voorbeeld gebruiken we bij het aanleggen van het woord `AWNING` de letter `W` van het woord `WEB` dat reeds op het bord ligt. Dit wordt genoteerd als `A(W)nING` (let erop dat we in de notatie met een kleine letter ook aangeven dat de letter n als blanco op het bord gelegd werd). Als er in het woord dat wordt aangelegd meerdere opeenvolgende letters zijn die reeds op het bord lagen, dan wordt deze groep letters samen tussen ronde haakjes gezet. We gebruiken dus bijvoorbeeld de notatie `(Bo)G(I)E` en niet `(B)(o)G(I)E`.

Opmerking voor wie het spelletje Scrabble kent: in deze opgave houden we geen rekening met bijkomende regels in verband met bestaande woorden, of de mogelijkheid dat bij het aanleggen van een woord ook secundaire woorden kunnen gevormd worden.

Definieer een klasse `Scrabble` die kan gebruikt worden om een Scrabble spelbord voor te stellen waarop woorden kunnen aangelegd worden. Deze klasse moet minstens de volgende methoden ondersteunen:

- Een initialisatiemethode waaraan de grootte $n \in \mathbb{N}$ van het spelbord moet doorgegeven worden. Hierbij moet gelden dat $1 \leq n \leq 26$. Indien dit niet het geval is, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldig spelbord`.
- Een methode `__str__` die een stringvoorstelling van het object teruggeeft die aangeeft welke woorden reeds op het spelbord aangelegd werden. Hierbij worden lege posities op het spelbord voorgesteld met een koppelteken, posities waarop een blokje met een letter gelegd werd met een hoofdletter, en positie waarop een blanco blokje gelegd werd met een kleine letter. Bekijk onderstaande voorbeeldsessie om de details van de stringvoorstelling af te leiden.
- Een methode `aanleggen` waaraan twee stringargumenten moeten doorgegeven worden die de plaats aanduiden waarop een woord op het spelbord moet aangelegd worden. Indien het woord niet op de aangegeven plaats kan aangelegd worden, dan mag het spelbord niet wijzigen en moet een `AssertionError` opgeworpen worden met de boodschap `woord kan niet aangelegd worden`. Anders moet het woord op het spelbord aangelegd worden, en moet de notatie teruggegeven worden die aangeeft welke letters als blanco op het bord liggen en welke letters reeds op het bord lagen.

Voorbeeld

```
>>> bord = Scrabble(7)
>>> print(bord)
-------
-------
-------
-------
-------
-------
-------
>>> bord.aanleggen('3C', 'WEB')
'WEB'
>>> print(bord)
-------
-------
--WEB--
-------
-------
-------
```

```
-------
>>> bord.aanleggen('C2', 'AWnING')
'A(W)nING'
>>> print(bord)
-------
--A----
--WEB--
--n----
--I----
--N----
--G----
>>> bord.aanleggen('4A', 'ZINC')
'ZI(n)C'
>>> print(bord)
-------
--A----
--WEB--
ZInC---
--I----
--N----
--G----
>>> bord.aanleggen('4A', 'ZINCoID')
'(ZInC)oID'
>>> print(bord)
-------
--A----
--WEB--
ZInCoID
--I----
--N----
--G----
>>> bord.aanleggen('G1', 'OXIDIZE')
'OXI(D)IZE'
>>> bord.aanleggen('6A', 'ENNUI')
'EN(N)UI'
>>> bord.aanleggen('A1', 'KLEZMER')
'KLE(Z)M(E)R'
>>> bord.aanleggen('E3', 'BOGIE')
'(Bo)G(I)E'
>>> print(bord)
K-----O
L-A---X
E-WEB-I
ZInCoID
M-I-G-I
ENNUI-Z
R-G-E-E
>>> bord.aanleggen('1A', 'KWYJIBO')
'(K)WYJIB(O)'
>>> print(bord)
KWYJIBO
L-A---X
E-WEB-I
ZInCoID
M-I-G-I
ENNUI-Z
R-G-E-E
>>> bord.aanleggen('3C', 'WEBLIKE')
Traceback (most recent call last):
AssertionError: woord kan niet aangelegd worden
>>> bord.aanleggen('D3', 'ECOUTE')
Traceback (most recent call last):
AssertionError: woord kan niet aangelegd worden
>>> bord.aanleggen('7A', 'RIVIERE')
Traceback (most recent call last):
AssertionError: woord kan niet aangelegd worden
>>> print(bord)
KWYJIBO
L-A---X
E-WEB-I
ZInCoID
M-I-G-I
ENNUI-Z
```

```
R-G-E-E

>>> bord = Scrabble(42)
Traceback (most recent call last):
AssertionError: ongeldig spelbord
```

Opgave 3: Molecuulformule (JavaScript) (10 pt)

Chemische formules omschrijven uit welke chemische elementen en in welke aantallen atomen een molecuul is opgebouwd. Vermoedelijk is de chemische formule van water een formule die de meeste mensen wel kennen: `H2O` (normaalgezien worden subscripts gebruikt voor de getallen, maar in deze opgave schrijven we ze gewoon tussen de symbolische voorstellingen van de chemische elementen). Deze formule geeft aan dat één watermolecule bestaat uit twee atomen waterstof (`H`) en één atoom zuurstof (`O`).

Complexere chemische formules kunnen ronde haakjes bevatten, die aangeven dat er aan het hoofdmolecuul meerdere kopieën gehecht zijn van het deelmolecuul ingesloten tussen ronde haakjes. Zo heeft ijzer(III)sulfaat een chemische formule `Fe2(SO4)3`, die aangeeft dat één molecuul ijzer(III)sulfaat bestaat uit twee atomen ijzer (`Fe`), drie atomen zwavel (`S`) en twaalf atomen zuurstof (`O`). Hierbij dient het aantal atomen van de subformule `SO4` tussen ronde haakjes dus vermenigvuldigd te worden met drie.

Het doel van deze opgave bestaat erin om op basis van een gegeven chemische formule te bepalen uit welke chemische elementen en in welke aantallen atomen het molecuul is opgebouwd.

Opgave

In de chemie worden **elementen** symbolisch voorgesteld door een string die begint met een hoofdletter, gevolgd door nul of meer kleine letters (bv. `C` of `Fe`).

Een **chemische formule** is een string die bestaat uit een opeenvolging van groepen, waarbij elke groep één van volgende twee vormen kan aannemen:

- de symbolische voorstelling van een element, eventueel gevolgd door een getal $n \in \mathbb{N}$ dat aangeeft hoeveel atomen van het element in een molecuul voorkomen
- de chemische formule van een submolecuul, eventueel gevolgd door een getal $n \in \mathbb{N}$ dat aangeeft hoeveel kopieën van het submolecuul in een molecuul voorkomen

In beide gevallen is $n = 1$ indien de groep niet eindigt op een getal. Gevraagd wordt:

- Schrijf een functie `groepen` waaraan een chemische formule moet doorgegeven worden. De functie moet een array teruggeven met de opeenvolgende groepen van de gegeven chemische formule.
- Schrijf een functie `molecuulformule` waaraan een chemische formule moet doorgegeven worden. De functie moet een object teruggeven waarvan de sleutels gevormd worden door de symbolische voorstellingen van alle chemische elementen in een molecuul dat omschreven wordt door de gegeven formule. De corresponderende waarden geven aan hoeveel atomen van elk element voorkomen in een molecuul dat omschreven wordt door de gegeven formule.

Voorbeeld

```
>>> groepen("C6H1206");
["C6", "H12", "O6"]
>>> groepen("C4H8(OH)2");
["C4", "H8", "(OH)2"]
>>> groepen("PbCl(NH3)2(COOH)2");
["Pb", "Cl", "(NH3)2", "(COOH)2"]
>>> groepen("PbCl(NH3(H2O)4)2");
["Pb", "Cl", "(NH3(H2O)4)2"]

>>> molecuulformule("C6H1206");
{"O": 6, "H": 12, "C": 6}
>>> molecuulformule("C4H8(OH)2");
{"O": 2, "H": 10, "C": 4}
>>> molecuulformule("PbCl(NH3)2(COOH)2");
{"C": 2, "N": 2, "H": 8, "Pb": 1, "O": 4, "Cl": 1}
>>> molecuulformule("PbCl(NH3(H2O)4)2");
{"Cl": 1, "N": 2, "O": 8, "H": 22, "Pb": 1}
```

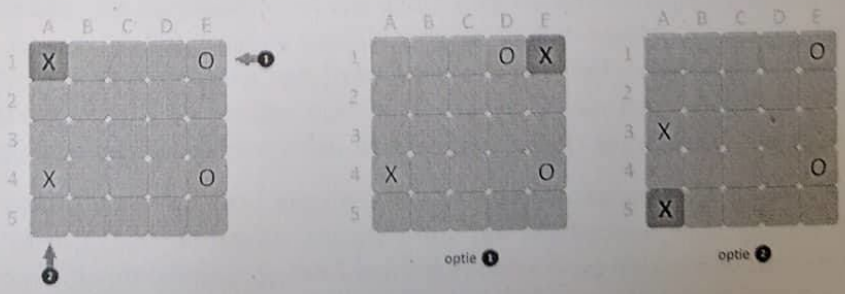
Opgave 4: Quixo (JavaScript) (10 pt)

Quixo is een bordspel dat gespeeld wordt met een vierkant $n \times n$ rooster van dobbelstenen. Elke dobbelsteen heeft minstens één zijde die blanco is, één zijde die gemarkeerd is met een kruis en één zijde die gemarkeerd is met een cirkel. Initieel liggen alle dobbelstenen met een blanco zijde naar boven. In het spel nemen twee spelers het tegen elkaar op, waarbij één speler speelt met de kruisen en de andere met de cirkels.

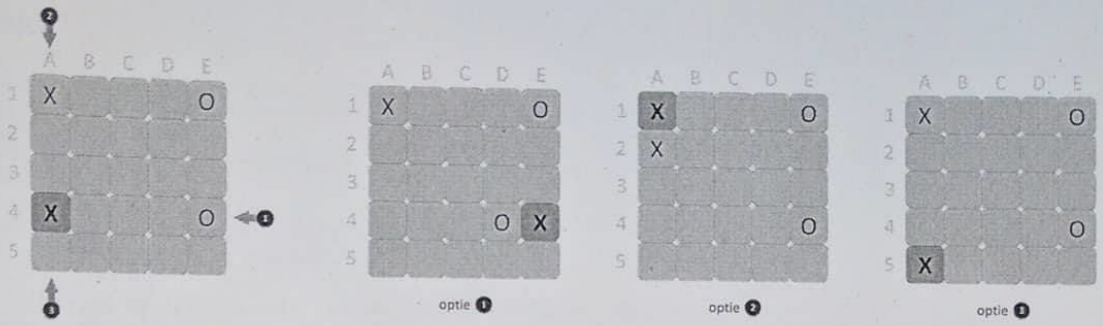
Om de beurt neemt een speler een dobbelsteen uit de buitenste rand van het rooster waarvan de bovenzijde blanco is of gemarkeerd is met zijn symbool. Indien de bovenzijde blanco was, draait hij de dobbelsteen zodat de bovenzijde gemarkeerd is met zijn symbool. Daarna duwt hij de dobbelsteen terug in het rooster aan het uiteinde van de rij of kolom waarop de dobbelsteen werd verwijderd, maar niet op dezelfde plaats waar de dobbelsteen werd verwijderd. Op die manier verwisselen per beurt telkens een paar dobbelstenen van plaats, en worden blanco dobbelstenen geleidelijk aan vervangen door kruisen en cirkels.

Het spel gaat verder totdat een speler erin slaagt om horizontaal, verticaal of diagonaal n dobbelstenen op één lijn te krijgen waarvan de bovenzijde telkens gemarkeerd is met zijn symbool. Die speler wint het spel. Als op hetzelfde moment lijnen gevormd worden met de symbolen van beide spelers, dan eindigt het spel op een gelijkspel.

Als je een dobbelsteen wegneemt in één van de hoeken van het rooster, dan zijn er dus twee mogelijkheden:



Als je een dobbelsteen aan de rand wegneemt die niet in een hoek van het rooster ligt, dan zijn er drie mogelijkheden:



Opgave

In deze opgave stellen we dobbelstenen met een blanco bovenzijde voor door een koppelteken (`-`), dobbelstenen waarvan de bovenzijde gemarkeerd is met een kruis door de letter `X`, en dobbelstenen waarvan de bovenzijde gemarkeerd is met een cirkel door de letter `O`.

Elke kolom in het rooster wordt van links naar rechts gemarkeerd met een volgende hoofdletter uit het alfabet (we veronderstellen dat er niet meer dan 26 kolommen zijn). Elke rij wordt van boven naar onder gemarkeerd met een getal, te beginnen vanaf 1. Op die manier kan elke positie in het rooster omschreven worden door een string die bestaat uit een hoofdletter (die de kolom aanduidt) en een getal (die de rij aanduidt).

Gevraagd wordt om een klasse `Quixo` te definiëren, waarmee het spelbord van een spelletje Quixo kan voorgesteld worden. Bij het instantiëren van objecten van de klasse `Quixo` moet een getal $n \in \mathbb{N}_0$ doorgegeven worden dat aangeeft hoeveel rijen en kolommen van dobbelstenen er zijn in het spelbord. Na initialisatie liggen alle dobbelstenen standaard met hun blanco zijde naar boven, maar optioneel kan er bij het instantiëren ook nog een string opgegeven worden die bestaat uit n^2 karakters `-`, `X` of `O`. Deze string omschrijft de bovenzijde van de dobbelstenen bij initialisatie als het rooster van links naar rechts en van boven naar onder wordt overlopen. Als er geen geldige argumenten doorgegeven worden bij het instantiëren van objecten van de klasse `Quixo`, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldig rooster`. Voorts moet de klasse `Quixo` minimaal de volgende methoden ondersteunen:

- Een methode `toString` waaraan geen argumenten moeten doorgegeven worden. De methode moet een stringvoorstelling van het rooster teruggeven, waarbij elke rij van het rooster op een afzonderlijke regel staat, en elke rij wordt voorgesteld als een opeenvolging van de karakters die de bovenzijde van de dobbelstenen voorstellen (`-`, `X` en `O`).
- Een methode `positie` waaraan de omschrijving van een positie aan de rand van het rooster moet doorgegeven worden. De methode moet een array met twee natuurlijke getallen teruggeven, die respectievelijk de rij- en kolomindex van de positie in het rooster aangeven als de rijen van boven naar onder en de kolommen van links naar rechts geïndexeerd worden vanaf nul. Indien de gegeven omschrijving geen geldige positie aan de rand van het rooster aanduidt, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldige positie`.
- Een methode `beurt` waaraan drie argumenten moeten doorgegeven worden: *i*) het symbool van een speler (`X` of `O`), *ii*) de positie aan de rand van het rooster waar een dobbelsteen wordt weggenomen en *iii*) de positie aan de rand van het rooster waar de dobbelsteen terug wordt ingeschoven. De methode moet de toestand van de dobbelstenen op het spelbord aanpassen volgens de beurt die wordt omschreven door de argumenten die aan de methode worden doorgegeven. De methode moet een referentie teruggeven naar het object waarop de methode werd aangeroepen. Indien de argumenten geen geldige beurt omschrijven, dan moet een `AssertionError` opgeworpen worden met de boodschap `ongeldige beurt`.

Voorbeeld

```
>>> const quixo01 = new Quixo(5, "X-XXX-XO-OO-OOOOXOX-OOOO-");
>>> quixo01.toString();
"X-XXX\n-XO-O\nO-OOO\nOXOX-\nOOOO-"
>>> quixo01.positie('B1');
[0, 1]
>>> quixo01.beurt('X', 'B1', 'B5').toString();
"XXXXX\n--O-O\nOXOOO\nOOOX-\nOXOO-"

>>> const quixo02 = new Quixo(5, "X-XXX-XO-OO-OOOOXOX-OOOO-");
>>> quixo02.toString();
"X-XXX\n-XO-O\nO-OOO\nOXOX-\nOOOO-"
>>> quixo02.beurt('X', 'B1', 'A1').toString();
"XXXXX\n-XO-O\nO-OOO\nOXOX-\nOOOO-"

>>> const quixo03 = new Quixo(5, "X-XXX-XO-OO-OOOOXOX-OOOO-");
>>> quixo03.toString();
"X-XXX\n-XO-O\nO-OOO\nOXOX-\nOOOO-"
>>> quixo03.beurt('X', 'B1', 'E1').toString();
"XXXXX\n-XO-O\nO-OOO\nOXOX-\nOOOO-"
```

De simulatie van de drie beurten uit bovenstaande voorbeeldsessie worden hieronder grafisch voorgesteld.